SD500069

Forge Viewer＋Design Automation AutoCAD で作るコンフィギュレータ

伊勢崎俊明
オートデスク株式会社

学習の目的

- Forge を利用する Web アプリとデスクトップ版 CAD の用途の違いを理解する
- Forge Design Automation API for AutoCAD を理解する
- デジタル トランスフォーメーション実現に向けた Forge 活用のヒントを把握する
- AutoCAD アドインのさらなる活用方法を見出す

説明

AutoCAD で作成した 3D モデル（DWG）を Forge Viewer に活用したコンフィギュレータを Web アプリとして作成、入力指示されたパラメータから、見積書や製作指示書を PDF 生成する例をご紹介します。使用したアイデアや実装に必要な要素をご案内するだけではなく、関連ワークフローをつなげていくアイデアとしてご覧いただけます。単なる図面作成だけではなく、デジタル トランスフォーメーション実現のヒントにしていただけるはずです。

スピーカーについて

伊勢崎 俊明

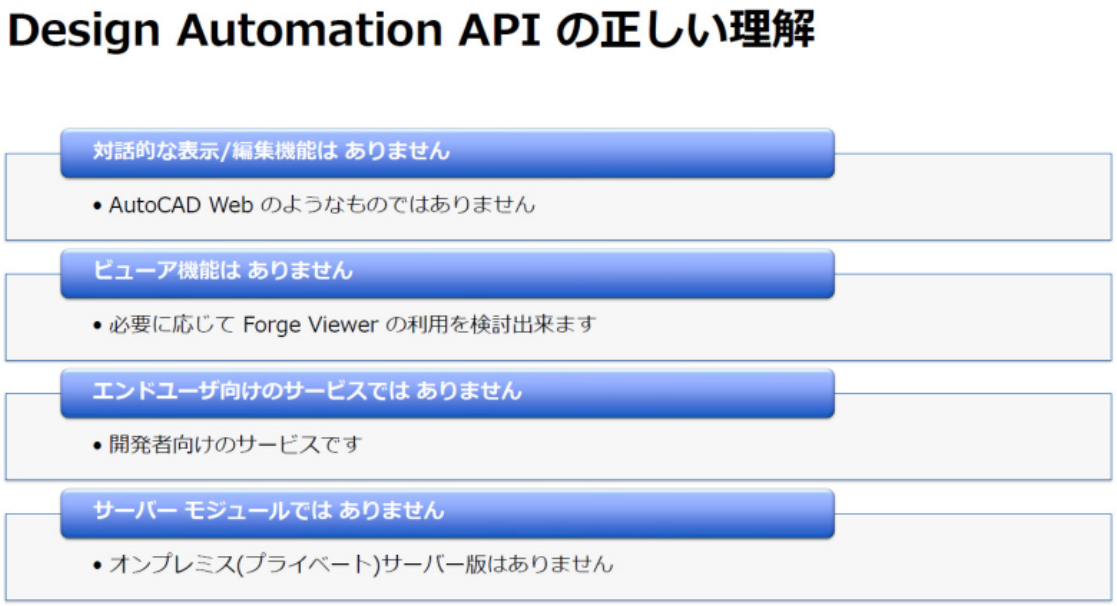
Chuo-ku, Tokyo, JP

Developer Advocacy & Support
Autodesk Japan
Software Development

CAD / CAM 業界の様々な企業での製品教育、製品/開発サポート、アプリケーション開発を経て、現在、オートデスクに 20 年超在籍。セールス エンジニアを経て、過去から最新までの技術変遷を踏まえて、Forge を啓蒙するエバンジェリストとして活動中。

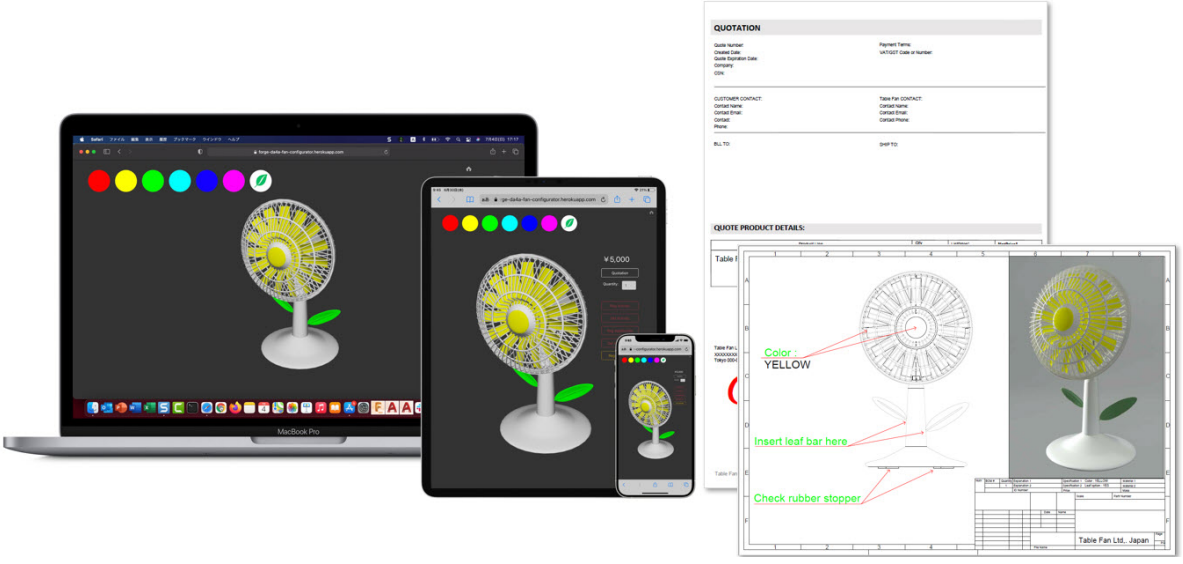
Design Automation API とは

Design Automation API は、クラウド上でユーザ インタフェースのない軽量なコアエンジン（AutoCAD、Revit、Inventor、3ds Max）を実行させて、アドインをロード、実行することで、ファイルなどの成果物を得る、といった API です。



クラスの目標

このクラスでは、煩雑さを避ける方向で効果的な AutoCAD で作成したモデルを用いた卓上扇風機のコンフィギュレータを作成していきたいと思います。コンフィギュレーターでは、オンライン販売ページを想定して、扇風機の羽の色、葉っぱ装飾の有無、扇風機の数を指定するものとして、成果物として、見積り書と製作指示書を PDF で生成するものとします。



まず、esign Automation API を利用する際には、次の概念を理解する必要があります。

Activity（アクティビティ）
Design Automation API に処理させるジョブ（実行）単位で、後述の WorkItem に関連付けられて指定されます。

WorkItem（ワークアイテム）
Design Automation API で扱う処理単位で、関連付けされた Activity の内容を実行します。Activity によっては WorkItem 側で Activity 用の入力パラメータと出力パラメータを指定します。パラメータには、通常、編集対象の既存 DWG ファイルの入手先 URL と、編集後に DWG ファイルを保存するための URL を指定します。

AppBundle（App バンドル）
Design Automation API に渡して、実行させるアドイン アプリケーションです。ObjectARX または .NET API、AutoLISP（除く Visual LISP 関数）で作成されて、自動ローダーでロード処理の詳細を提供する必要があります。自動ローダーとは、**PackageContents.xml ファイル**を利用して AutoCAD へのアドイン ロードを実行する仕組みです。

Engine（エンジン・コアエンジン）
クラウド上で稼働し、WorkItem を処理する AcCoreConsole.exe です。エンジンには、AutoCAD バージョンに準じた AcCoreConsole.exe をエンジンバージョンとして指定して実行させることが出来ます。

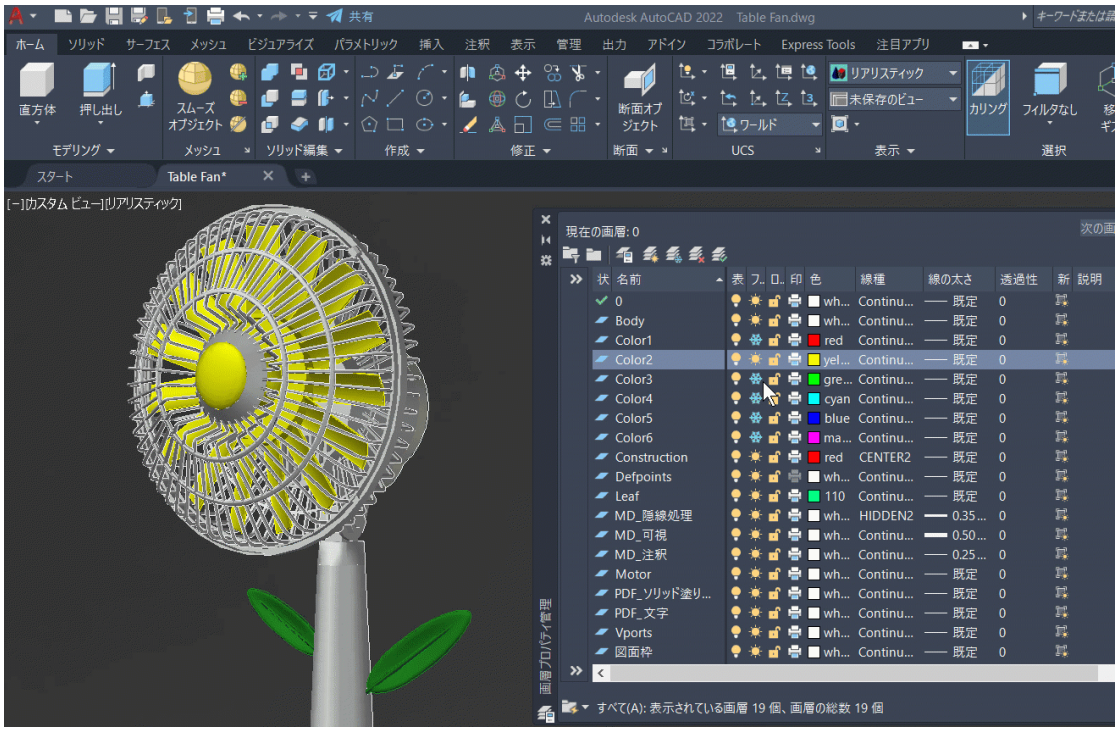
コンフィギュレータの考察：ForgeViewer 表現編

Design Automation API にはユーザ インタフェースはありませんので、適宜、Forge Viewer と組み合わせることで、コアエンジンが生成した成果ファイルを Model Derivative API で SVF/SVF2 変換して、確認用に表示するのが一般的です。

ただ、コンフィギュレータの性格上、Design Automation API にパラメータを渡す前に Viewer 上で指定した内容を確認したい場合があります。このような場合、Forge Viewer のベースになっている three.js も機能（API）を利用たビューア上の脚色も可能ですが、コード量が増えてメインテナンスも負荷になる可能性があります。そこで、SVF / SVF2 変換する前の DWG の段階で、データに仕込みを組み込むことにします。

具体的には、画僧毎に色の異なる羽根を用意して、画僧のフリーズをオン、オフすることで色指定時のビューア上の変化を得られるようにしておきます。同様に、葉っぱ形状の有無の専用の画僧でオン、オフするものとします。

この時、羽根の色は、BYLAYER 色を使用します。なぜなら、Model Derivative API で DWG の 3D モデルを変換すると、AutoCAD で指定したマテリアルが欠落してしまう仕様があるためです。



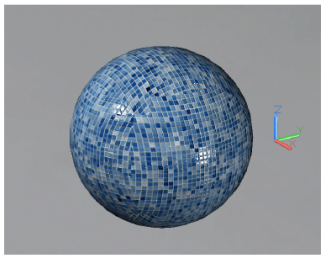
このような 3D モデル（DWG）を SVF/SVF2 変換した場合、次のようなコードで Forge Viewer 側の画僧のオン、オフをコントロールできます。

```
function setLayerVisibility(name, flag) {
    var root = _viewer.impl.getLayersRoot();
    if (root == null) {
        return;
    }
    for (var i = 0; i < root.childCount; i++) {
        var layer = root.children[i];
        if (layer.name === name) {
            _viewer.setLayerVisible([layer], flag, false);
        }
    }
}
```

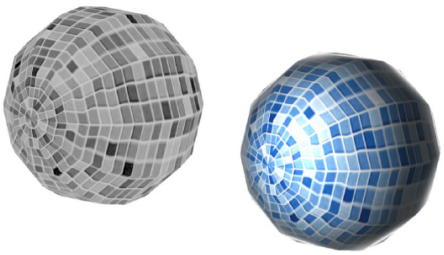
ただ、やはり CAD 上のマテリアルを指定したい場合もあります。ここでも three.js レベルでの実装が可能ですが、UV ラップなど、マテリアル パターンの原点や尺度を一致させるのに難儀します。

CAD 上のマテリアル表現との整合性

- CAD 上ではテクスチャ画像の調整が可能
  - 尺度
  - 位置
  - 回転角度

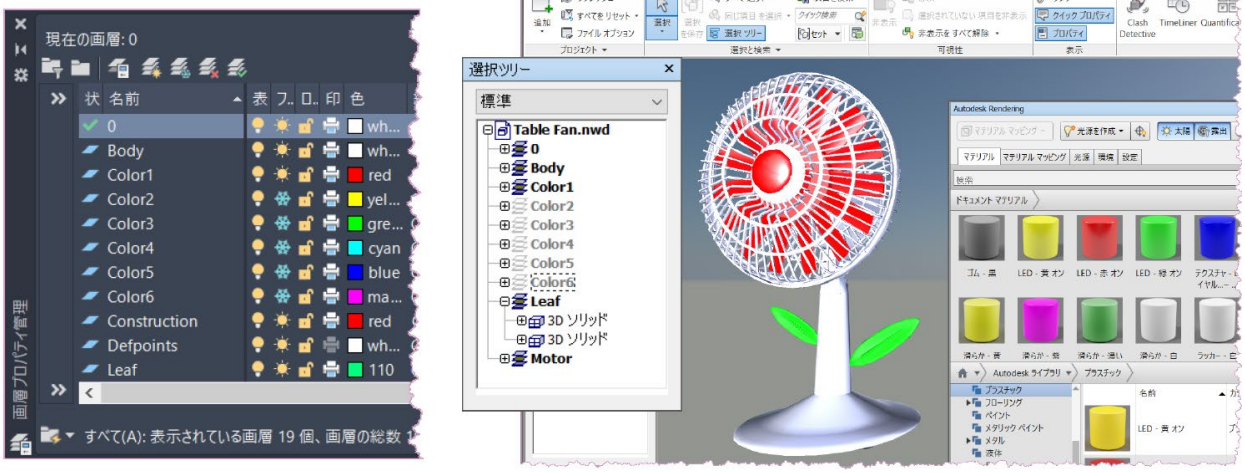


- three.js マテリアル表現との調整に難
  - CAD 製品のマテリアル表現との整合調整
  - どのポリゴンにどのマテリアルを適用するか？
  - シーン光源からの影響はどうするか？
  - プログラムによる調整が必須！



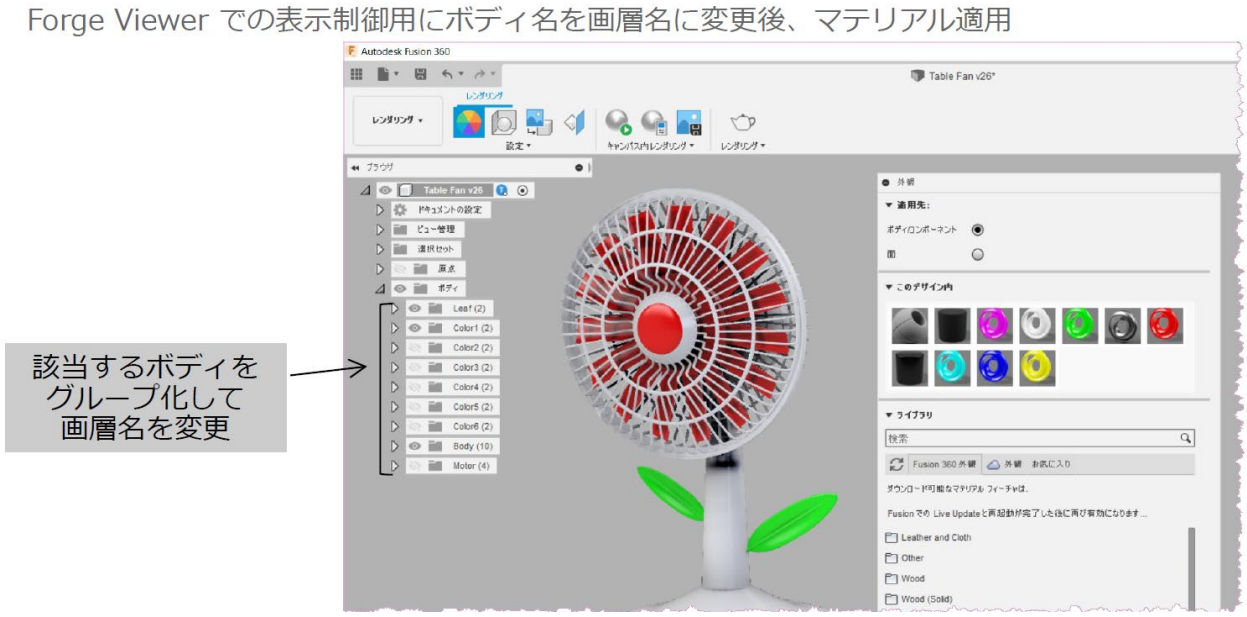
DWG をインポート出来て、SVF/SVF2 変換してもマテリアル情報を維持できるものに、Navisworks があります。Navisworks は AutoCAD の画僧情報も維持した状態で SVF/SVF2 化出来るので一石二鳥です。Forge Viewer 側の画僧のオン、オフのコードも流用出来ます。

Forge Viewer での画層表示制御はそのまま流用が可能、マテリアルは新規に適用



ただ、ここで問題があります。Navisworks はインフラなどの比較的大きなサイズのモデルをポリゴン化して取り込むように最適化されているので、今回の卓上扇風機のようなサイズだと、ビューア上のモデルの精緻さが失われてしまいます。

代替として次に考えられるのは、DWG を Fusion 360 にインポート、マテリアルを割り当てて、アーカイブ ファイル（.f3d）を SVF/SVF2 変換する方法があります。



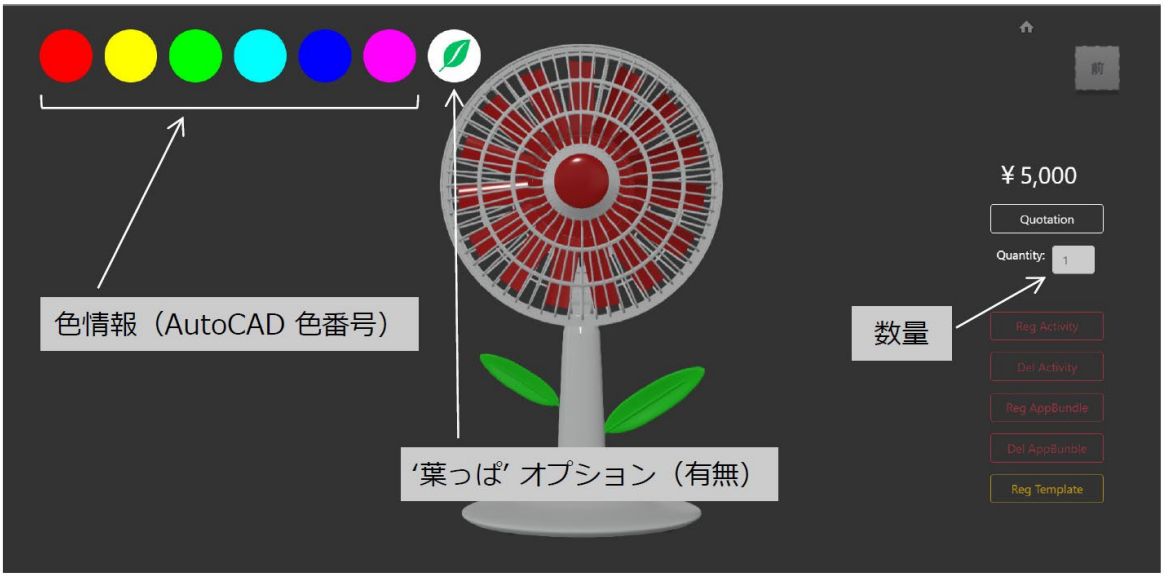
Fusion 360 上には画僧というコンセプトは存在しないので、インポートしたボディトに、色毎に異なる名前を指定して画僧の代わりをさせることにします。

もちろん、Forge Viewer 上のモデルのオン、オフのコードも書き換える必要があります。

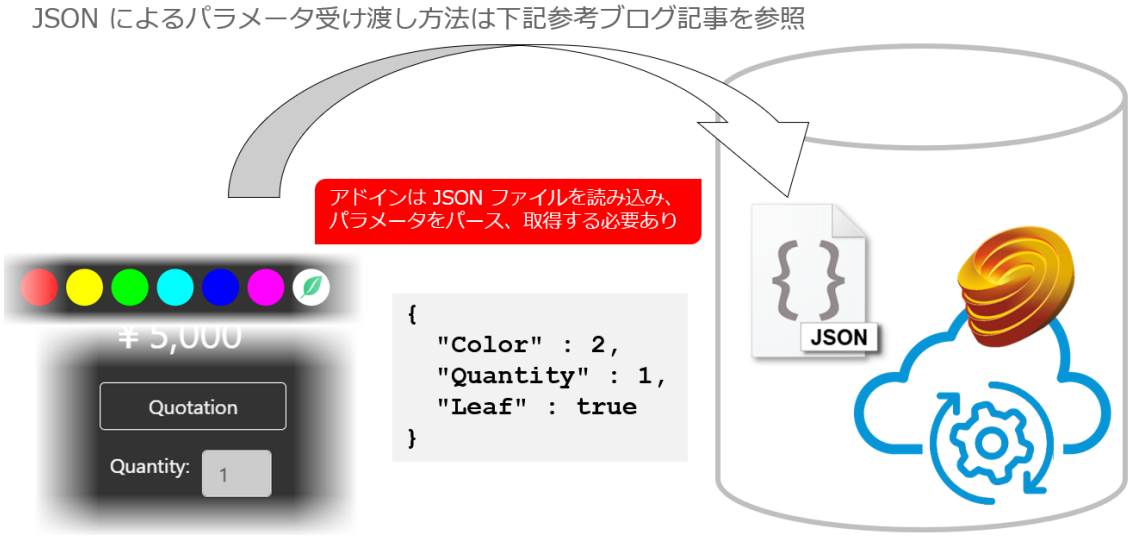
```
function setModelVisibility(name, flag) {
    _viewer.search(name, function (idArray) {
        if (flag) {
            _viewer.show(idArray);
        } else {
            _viewer.hide(idArray);
        }
    });
}
```

コンフィギュレータの考察：Design Automation API 処理編

ビューア画面からパラメータとして Design Automation API に渡すパラメータは、羽根の色、葉っぱの有無。注文する卓上扇風機の数です。



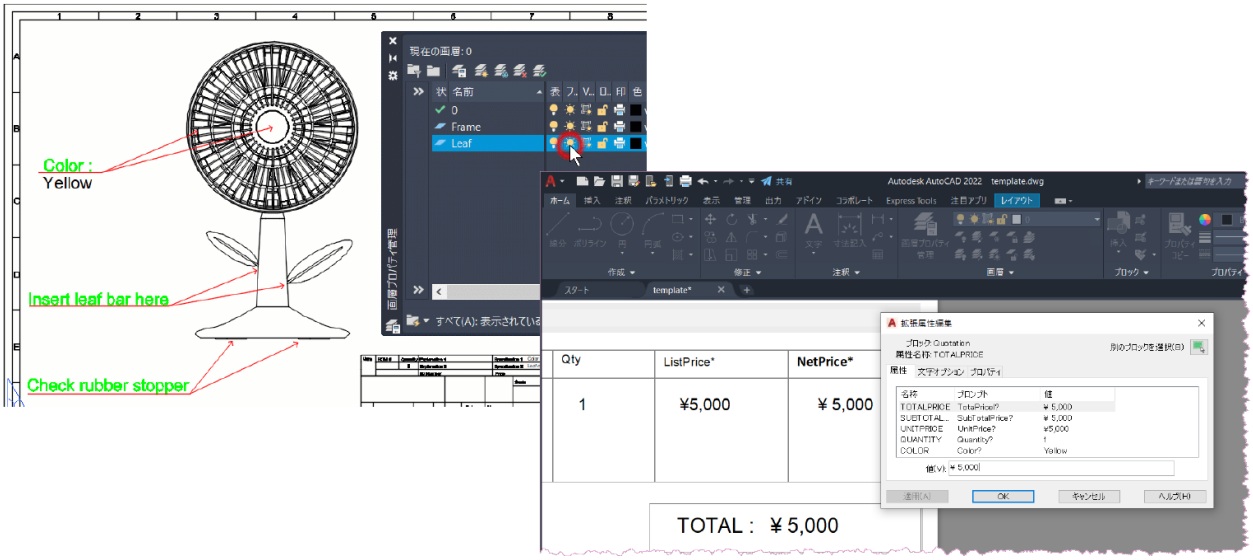
これらパラメータは。JSON データとして渡されて、コエンジンの実行環境に JSON ファイルとして保存されます。コアエンジンにロード、実行されるアドインは、その JSON ファイルを読み込んでパラメータをアドイン処理に反映させる必要があります。



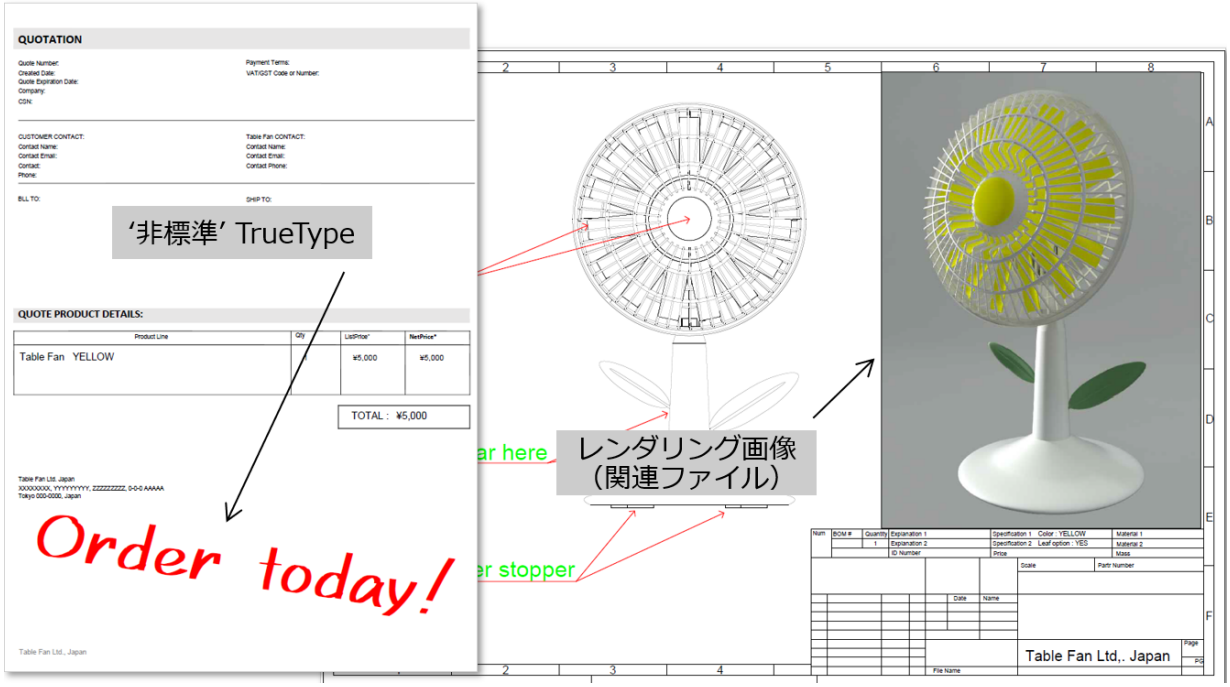
- https://adndevblog.typepad.com/technology_perspective/2020/07/forge-online-design-automation-api-for-autocad.html
- https://adndevblog.typepad.com/technology_perspective/2021/07/autocad-addin-conversion-to-design-automation-api.html

パラメータを反映させる見積書と製作指示書は、あらかじめブロック属性を仕込んだ素材ファイル DWG です。.NET アドインによる属性の書き換え処理は AutoCAD のオンライン ヘルプにある内容を踏襲することが出来ます。

素材ファイルのブロック属性とレイアウト上の画層オン/オフ制御で対応（AppBundle）



このクラスでご紹介しているコンフィギュレータでは、見積書にカスタムな TrueType フォントを利用します。同様に、製作指示書にあらかじめ用意したレンダリング画像を貼り付けます。TrueType フォントもレンダリング画像も。AppBundle に同梱してコアエンジン環境に渡すことが出来ます。



※参考フォント出典：https://fontfree.me/ 851 マカポップ Ver 0.01 フォント

AppBundle に TrueType フォント ファイルを同梱すると、Design Automation API はそのファイルを自動的に展開しますが、フォント名とフォントファイル名が一致しないと、コアエンジンが正しく TrueType フォントを認識することが出来ません。

これを解決するのは、フォント マッピング ファイル（.fmp）を利用して解決することが出来ます。

- Activity で登録、実行時に WorkItem で作業領域にダウンロード
  - 素材 DWG などと同様の扱い

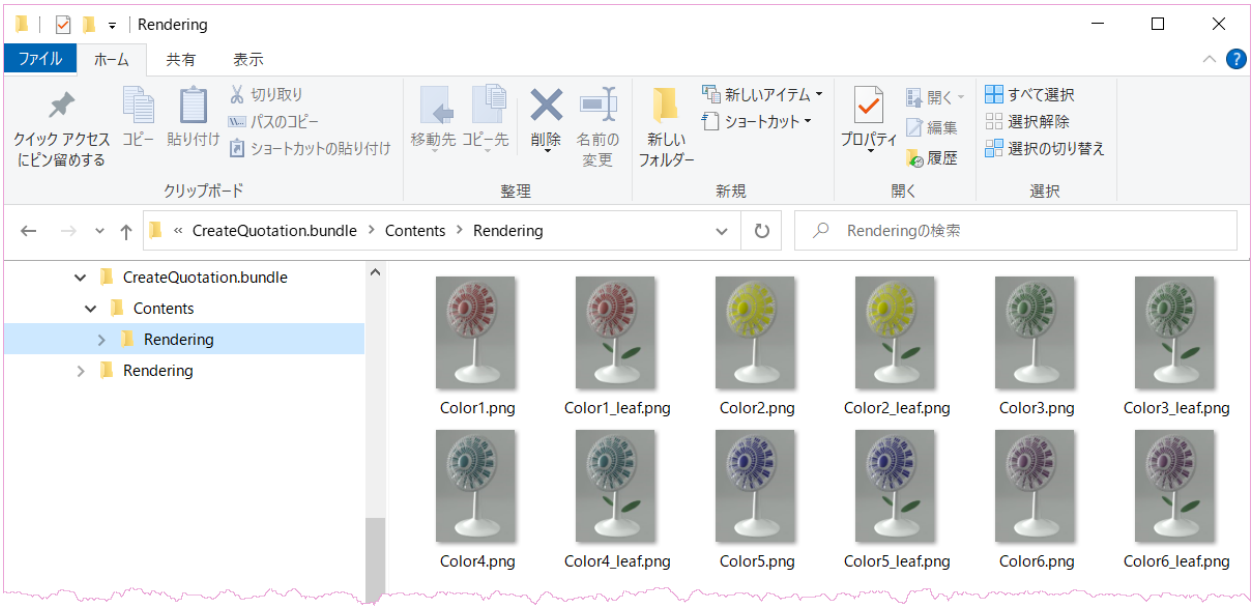

```
省略
"FontMap": {
    "zip": false,
    "ondemand": false,
    "verb": "get",
    "description": "Input font map file",
    "required": true,
    "localName": "dwg.fmp"
},
省略
```

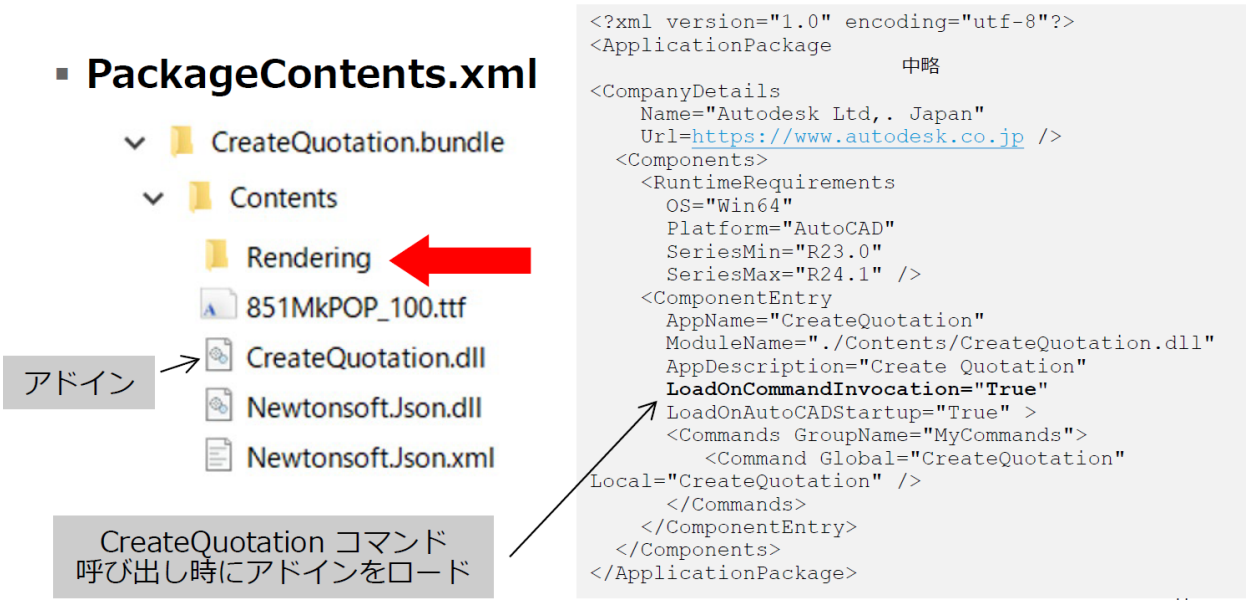
Avtivity 登録 JSON ボディ

```
省略
"FontMap": {
    "url": signedURLforInput2,
    "headers": {
        "Authorization": "Bearer " + access_token,
        "Content-type": "text/plain"
    },
    "verb": "get"
},
省略
```

WorkItem JSON ボディ

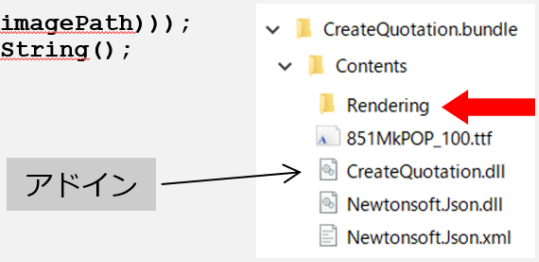
また、AppBundle に同梱したレンダリング画像をアドインからパス解決して貼り付けるには、次のような AppBundle 階層とアドインコードが必要です。





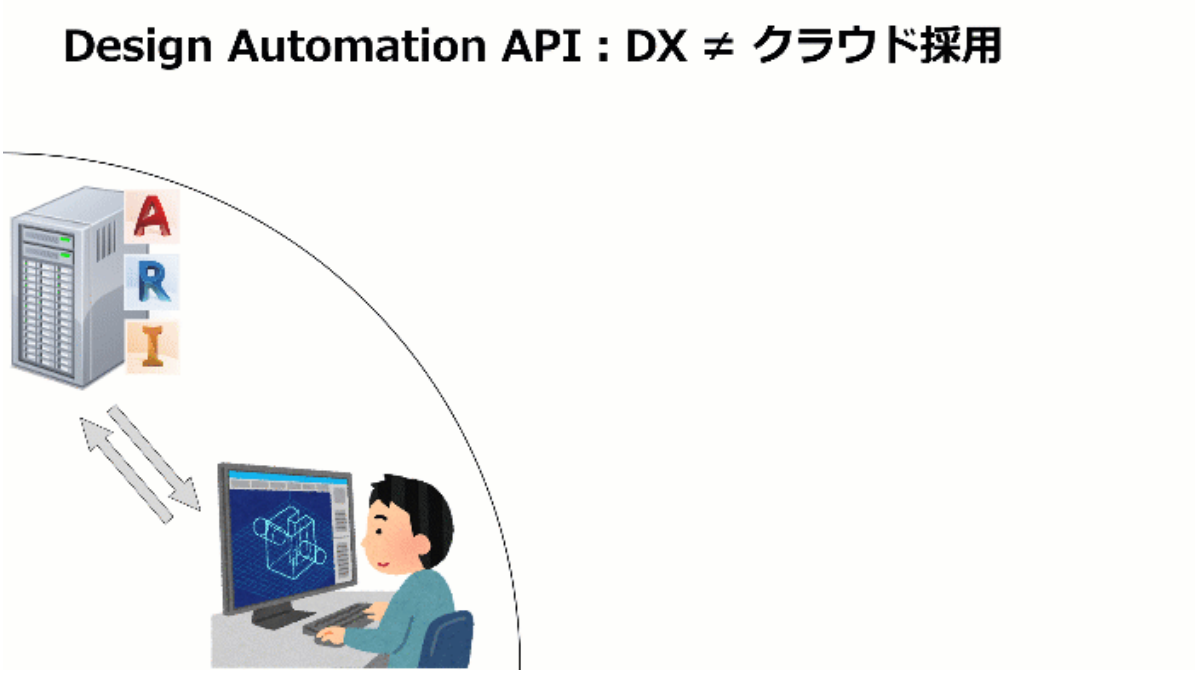
画像ファイルのパスはアドイン位置と AppBundle フォルダ構成から特定：C#（.NET API）

```
// Atttach rendered image
string imagePath = System.Reflection.Assembly.GetExecutingAssembly().Location;
imagePath = imagePath.Substring(0,
imagePath.IndexOf(System.IO.Path.GetFileName(imagePath)));
imagePath += "Rendering¥¥Color" + intColor.ToString();
if (boolLeaf)
{
    imagePath += "_Leaf.png";
}
else
{
    imagePath += ".png";
}
bool result = AttachImage(imagePath, "Instruction", new Point3d(184.0, 52.0, 0.0),
105);
```

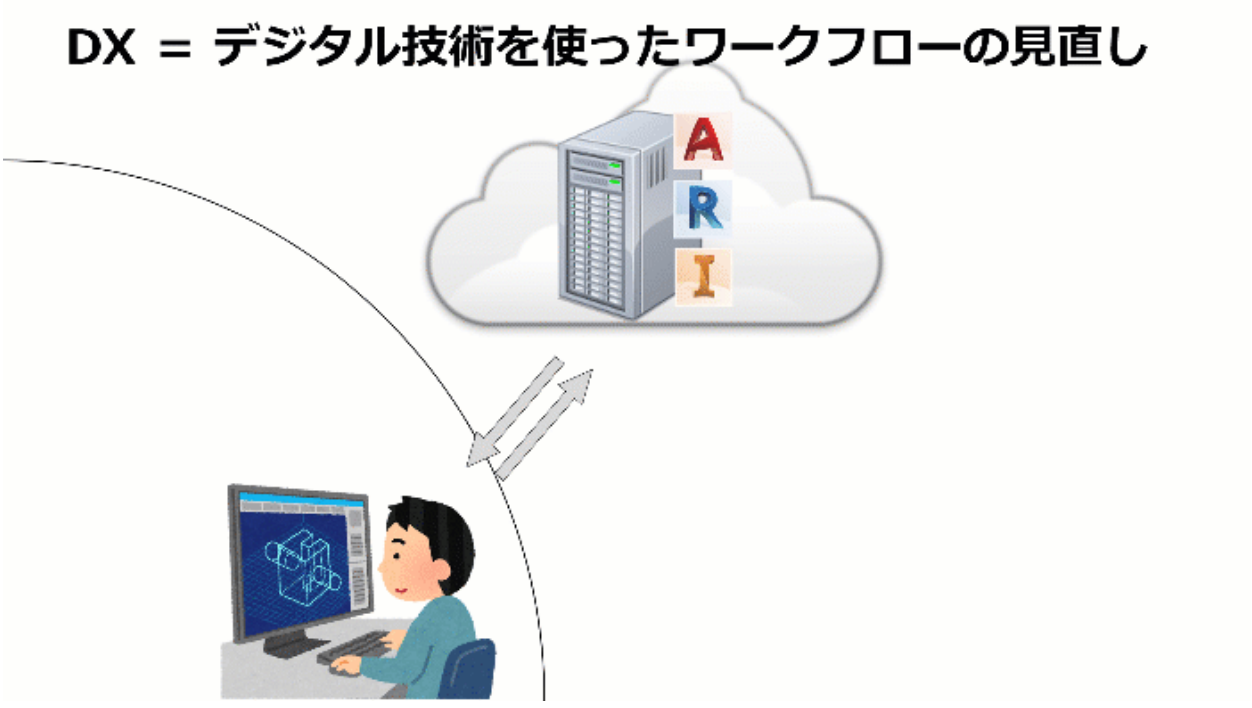


デジタルトランスフォーメーションを成功させるために

Design Automation API for AutoCAD を利用する利点とは何でしょう？
Web・クラウド時代に生産性向上や自動化を目指すにあたり、オンプレミス環境（社内サーバー）の AutoCAD 利用から、クラウドで稼働する Design Automation API 利用に切り替えただけでは、デジタル トランスフォーメーションの利点は、あまり感じられないかもしれません。

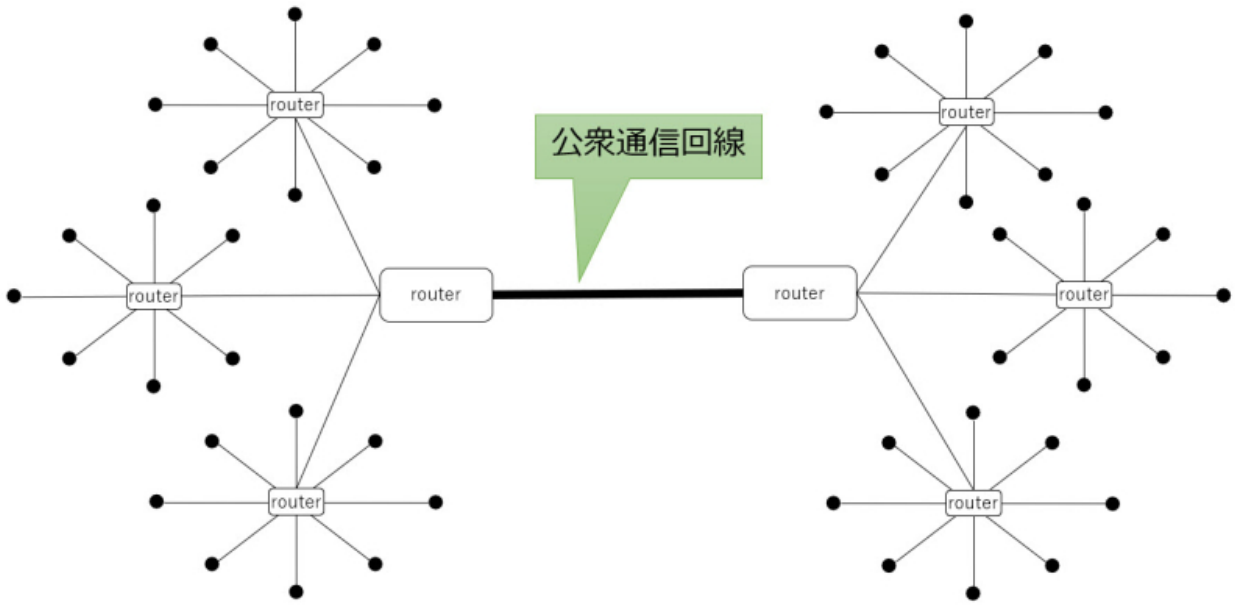


一律に論じることは出来ませんが、デジタル トランスフォーメーションの成果は、デジタル技術を利用したワークフローの見直しを目標としたほうが、新しい価値を見出せるように感じます。例えば、Design Automation API で得られた成果物をダウンロードして紙に印刷・出図手続きを経て社外とコミュニケーションする、といった従来のものではなく、デジタルに配信、コラボレーションするような発展的思想を同時に導入する、といった具合です。

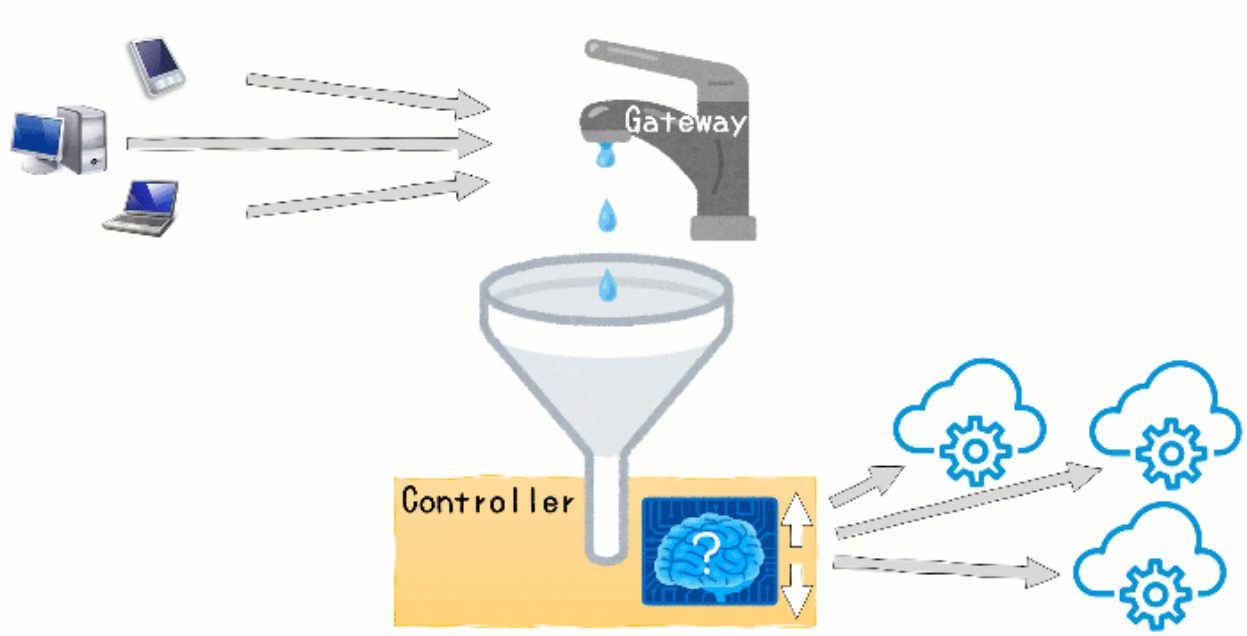


さて、この Design Automation API 環境は、他の Forge API と同じくオートデスクが AWS に構築している開発プラットフォームです。Forge を利用するアプリケーション（Forge アプリ）は、通常、Web サーバーにホストされる Web アプリとして動作することになります。この場合、Forge アプリを利用するユーザは、クライアントとなるデバイスから Web ブラウザを使って様々なリクエストを送信し、Forge アプリからのレスポンスを得ながら特定のタスクを実行していくことになります。

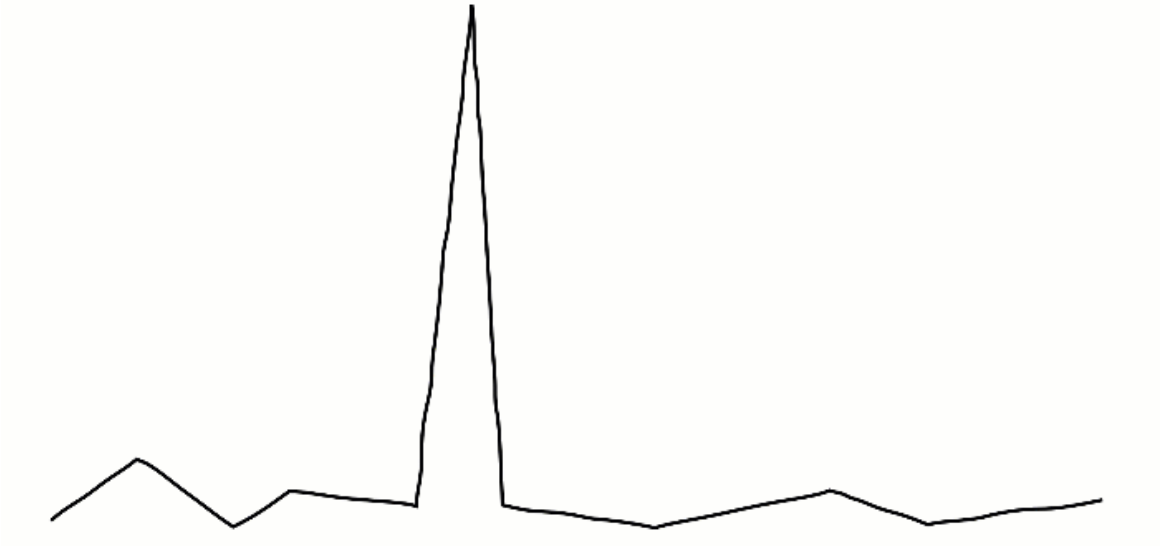
この際、伝送路として使用するのは、言うまでもなく、公衆回線を使ったインターネットです。このため、「今日はページの表示が早い」、「昨日より遅い」、など、アクセスする時間帯や状況によって通信に時間がかかってしまう可能性が考えられます。このようなインターネット通信によるレスポンスの差は、結果として、Design Automation API の Web アプリ部分でパフォーマンス差として現れることになってしまいます。



もう一つ、アドイン部分の実行環境にも、ご承知いただきたい点があります。
Design Automation API のアドイン処理も含め、CPU リソースを利用する演算サービスの場合、クラウドの自動伸張機能（elastic computing）の特性も理解していただきたい点です。具体的には、Design Automation API は、リクエスト数に応じて、実行環境となる仮想マシンを動的に増減します。



もし、非常に短い間に数多くにリクエストが集中した場合（スパイク）、必要数の仮想マシン展開に時間がかかり、処理完了までに「通常」より時間を要してしまう場合もあります。オートデスクは、もちろん、このような遅延を低減するための投資を続けています。



パブリック クラウドのインフラ（AWS）上に構築されている Forge は、世界中の Forge デベロッパとの共有リソースであるとの認識も必要かと思います。（もちろん、ストレージ内容が他者から勝手に見られてしまう、という意味ではありません。）
デジタルトランスフォーメーション成功のカギは、ワークフローの見直しから生まれる新しい価値の創出・発見にあるはずです。そんな観点で運用を捉えていただければと思います。

まとめ

- Forge Viewer 表現編
  - CAD データの特性（画層、ボディ名）を生かして表示制御
  - DWG 改め NWD 改め F3D
- Design Automation API 処理編
  - 事前準備したデータ（フォント、画像）で成果データをリッチ化
- 素材データと成果データは同じ形式である必要はない
  - コンフィギュレータ ビューアは 3D 表現優先でも OK
  - ブロック属性を仕込んだ DWG を PDF 化